INTRODUCTION TO MACHINE LEARNING

HIDDEN MARKOV MODELS

* The contents are adapted from Dr. Jean Gao at UT Arlington

Mington Kang, Ph.D.
Department of Computer Science @ UNLV
given say a T in our input sequence, which state emitted it?
Hidden State

- we’ll distinguish between the observed parts of a problem and the hidden parts
- in the Markov models we’ve considered previously, it is clear which state accounts for each part of the observed sequence
- in the model above, there are multiple parts that could account for each part of the observed sequence
  - this is the hidden part of the problem
Observed: Lab, Coffee, Lab, Coffee, Lab, Lab, Bar, Bar, Coffee, Bar, Coffee
Hidden: H, H, H, H, H, H, D, D, D, D, D
The Parameters of an HMM

as in Markov chain models, we have transition probabilities

\[ a_{kl} = \Pr(\pi_i = l \mid \pi_{i-1} = k) \]

probability of a transition from state \( k \) to \( l \)

\( \pi \) represents a path (sequence of states) through the model

since we’ve decoupled states and characters, we might also have emission probabilities

\[ e_k(b) = \Pr(x_i = b \mid \pi_i = k) \]

probability of emitting character \( b \) in state \( k \)
A Simple HMM

$a_{13}$ probability of a transition from state 1 to state 3

$e_2(A)$ probability of emitting character $A$ in state 2
Simple HMM for Gene Finding

Figure from A. Krogh, An Introduction to Hidden Markov Models for Biological Sequences
Three Important Questions

- How likely is a given sequence?
  the Forward algorithm

- What is the most probable “path” for generating a given sequence?
  the Viterbi algorithm

- How can we learn the HMM parameters given a set of sequences?
  the Forward-Backward (Baum-Welch) algorithm
How Likely is a Given Sequence?

- the probability that a **CERTAIN** path \( \pi_0 \ldots \pi_N \) is taken and the sequence \( x_1 \ldots x_L \) is generated:

\[
Pr(x_1 \ldots x_L, \pi_0 \ldots \pi_N) = a_{\pi_0 \pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}}
\]

(assuming begin/end are the only silent states on path)
How Likely Is A Given Sequence?

Pr(AAC, π) = a_{01} \times e_1(A) \times a_{11} \times e_1(A) \times a_{13} \times e_3(C) \times a_{35}
= 0.5 \times 0.4 \times 0.2 \times 0.4 \times 0.8 \times 0.3 \times 0.6
How Likely is a Given Sequence?

- the probability over all paths is:

\[
\Pr(x_1 \ldots x_L) = \sum_{\pi} \Pr(x_1 \ldots x_L, \pi_0 \ldots \pi_N)
\]

- but the number of paths can be exponential in the length of the sequence...
- the Forward algorithm enables us to compute this efficiently
How Likely is a Given Sequence: The Forward Algorithm

- define $f_k(i)$ to be the probability of being in state $k$ having observed the first $i$ characters of $x$

$$f_k(i) = P_{r}(x_1, x_2, ..., x_i, \pi_i = k)$$

- we want to compute $f_N(L) = P_r(x_1, x_2, ..., x_L)$ the probability of being in the end state having observed all of $x$ can define this recursively
The Forward Algorithm

- because of the Markov property, don’t have to explicitly enumerate every path – use dynamic programming instead

- e.g. compute $f_4(i)$ using $f_2(i-1), f_4(i-1)$
The Forward Algorithm

- initialization:

\[ f_0(0) = 1 \]

probability that we’re in start state and have observed 0 characters from the sequence

\[ f_k(0) = 0, \quad \text{for } k \text{ that are not silent states} \]
The Forward Algorithm

- recursion for emitting states ($i = 1 \ldots L$):

$$f_l(i) = e_l(x_i) \sum_k f_k(i-1) a_{kl}$$

- recursion for silent states:

$$f_l(i) = \sum_k f_k(i) a_{kl}$$
The Forward Algorithm

- **termination:**

\[
Pr(x) = Pr(x_1...x_L) = f_N(L) = \sum_k f_k(L)a_{kN}
\]

probability that we’re in the end state and have observed the entire sequence
given the sequence $x = \text{TAGA}$
Forward Algorithm Example

- given the sequence \( x = \text{TAGA} \)
- initialization

\[
\begin{align*}
f_0(0) &= 1 \quad f_1(0) = 0 \quad \ldots \quad f_5(0) = 0
\end{align*}
\]

- computing other values

\[
\begin{align*}
f_1(1) &= e_1(T) \times (f_0(0) \times a_{01} + f_1(0) a_{11}) = \\
&= 0.3 \times (1 \times 0.5 + 0 \times 0.2) = 0.15
\end{align*}
\]

\[
\begin{align*}
f_2(1) &= 0.4 \times (1 \times 0.5 + 0 \times 0.8) = 0.2
\end{align*}
\]

\[
\begin{align*}
f_1(2) &= e_1(A) \times (f_0(1) \times a_{01} + f_1(1) a_{11}) = \\
&= 0.4 \times (0 \times 0.5 + 0.15 \times 0.2)
\end{align*}
\]

\[
\begin{align*}
\Pr(\text{TAGA}) &= f_5(4) = (f_3(4) \times a_{35} + f_4(4) a_{45})
\end{align*}
\]
in some cases, we can make the algorithm more efficient by taking into account the minimum number of steps that must be taken to reach a state.

- e.g. for this HMM, we don’t need to initialize or compute the values:
  \[ f_3(0), f_4(0), f_5(0), f_5(1) \]
Three Important Questions

- How likely is a given sequence?
- What is the most probable “path” for generating a given sequence?
- How can we learn the HMM parameters given a set of sequences?
Finding the Most Probable Path: The Viterbi Algorithm

- Define \( \nu_k(i) \) to be the probability of the most probable path accounting for the first \( i \) characters of \( x \) and ending in state \( k \)

\[
\nu_k(i) = \max_{\pi_1, \pi_2, \ldots, \pi_{i-1}} \Pr(\pi_1, \pi_2, \ldots, \pi_i = k; x_1, x_2, \ldots, x_i)
\]

- We want to compute \( \nu_N(L) \), the probability of the most probable path accounting for all of the sequence and ending in the end state
- Can define recursively
- Can use DP to find \( \nu_N(L) \) efficiently
Finding the Most Probable Path: The Viterbi Algorithm

- **Initialization:**

  \[ v_0(0) = 1 \]

  \[ v_k(0) = 0, \text{ for } k \text{ that are not silent states} \]
The Viterbi Algorithm

- recursion for emitting states ($i = 1 \ldots L$):

$$v_l(i) = e_l(x_i) \max_k [v_k(i-1) a_{kl}]$$

$$p_{tr}(i) = \arg \max_k [v_k(i-1) a_{kl}] \quad \text{keep track of most probable path}$$

- recursion for silent states:

$$v_l(i) = \max_k [v_k(i) a_{kl}]$$

$$p_{tr}(i) = \arg \max_k [v_k(i) a_{kl}]$$
The Viterbi Algorithm

- termination:

\[
\Pr(x, \pi) = \max_k \left( v_k(L)a_{kN} \right)
\]

\[
\pi_L = \arg \max_k \left( v_k(L)a_{kN} \right)
\]

- traceback: follow pointers back starting at \( \pi_L \)
Forward & Viterbi Algorithms

- Forward/Viterbi algorithms effectively consider all possible paths for a sequence
  - Forward to find probability of a sequence
  - Viterbi to find most probable path
- Consider a sequence of length 4…
Three Important Questions

- How likely is a given sequence?
- What is the most probable “path” for generating a given sequence?
- How can we learn the HMM parameters given a set of sequences?
The Learning Task

given:
- a model
- a set of sequences (the training set)

do:
- find the most likely parameters to explain the training sequences

the goal is to find a model that generalizes well to sequences we haven’t seen before
Learning Parameters

- If we know the state path for each training sequence, learning the model parameters is simple:
  - No hidden state during training
  - Count how often each parameter is used
  - Normalize/smooth to get probabilities
  - Process just like it was for Markov chain models

- If we don’t know the path for each training sequence, how can we determine the counts?
  - Key insight: estimate the counts by considering every path weighted by its probability
Learning without Hidden State

- learning is simple if we know the correct path for each sequence in our training set

- estimate parameters by counting the number of times each parameter is used across the training set
if we don’t know the correct path for each sequence in our training set, consider all possible paths for the sequence

begin

end

estimate parameters through a procedure that counts the expected number of times each parameter is used across the training set
Learning Parameters:
The Baum-Welch Algorithm

- *a.k.a* the Forward-Backward algorithm
- an *Expectation Maximization* (EM) algorithm
  - EM is a family of algorithms for learning probabilistic models in problems that involve hidden state
- in this context, the hidden state is the path that best explains each training sequence
HMM for DNA sequence

(a) State transitions and emission probabilities:

1. State 1:
   - Transition to state 1 with probability 0.99
   - Emissions:
     - A with probability 0.4
     - C with probability 0.1
     - G with probability 0.1
     - T with probability 0.4

2. State 2:
   - Transition to state 2 with probability 0.9
   - Transition back to state 1 with probability 0.01
   - Emissions:
     - A with probability 0.05
     - C with probability 0.4
     - G with probability 0.5
     - T with probability 0.05

(b) State sequence (hidden):

... 1 1 1 1 1 2 2 2 2 1 1 ...

Transitions: ? 0.99 0.99 0.99 0.99 0.01 0.9 0.9 0.9 0.1 0.99

(c) Symbol sequence (observable):

... A T C A A G G C G A T ...

Emissions: 0.4 0.4 0.1 0.4 0.4 0.5 0.5 0.4 0.5 0.4 0.4

Eddy, “Hidden Markov models”, 1996
HMM for modeling eukaryotic genes

Yoon, “Hidden Markov Models and their Applications in Biological Sequence Analysis”, Current Genomics, 2009